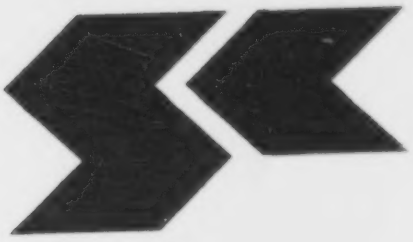
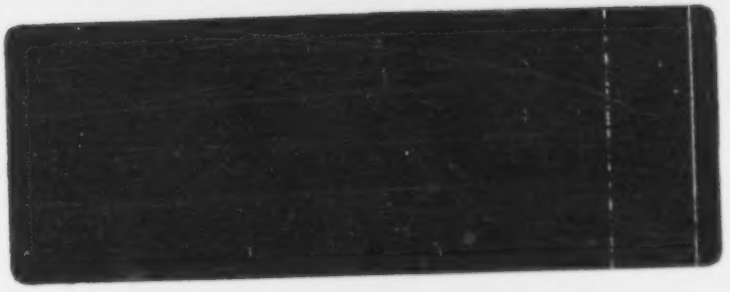
P1485509.PDF [Page: 1 of 35]

93-669





P148599.PDF [Page: 4 of 35]

ACES Ethernet Toolkit
Final Report

Submitted to: Defence Research Establishement Ottawa

Software Kinetics Document No. 1600-118-001 Version 01
Copy #2 31 March 1993

ACES ETHERNET TOOLKIT

FINAL REPORT

Contract No. W7714-02-9601

31 March 1993

Prepared for:

Mr. Derek Elsaesser
Defence Research Establishment Ottawa
Electronic Warfare Division
Ottawa, Ontario

Prepared by:

Software Kinetics Ltd.
65 Iber Road,
Stittsville, Ontario Canada
K2S 1E7

Software Kinetics Document No. 1600-118-001 *Version* 01

CAUTION
The use of this information is permitted subject to recognition of proprietary and patent rights.

P148599.PDF [Page: 6 of 35]

Document Approval Sheet

Document No.: 1600-118-001 *Version* 01

Document Name: ACES Ethernet Toolkit Final Report

Approvals	**Signature**	**Date**
Project Manager:	J. P. Perrin	31 mar 93

P148599.PDF [Page: 7 of 35]

Document Revision History

Revision	Description of Changes	Origin Date
01	New Document Issued	31 March 1993

P148599.PDF [Page: 8 of 35]

Table Of Contents

1. INTRODUCTION 1
1.1 Background 1

2. REFERENCES 3

3. INTERFACE DESCRIPTION 4
3.1 Ethernet Communications Interface 4
3.1.1 Stream Sockets 4
3.1.2 Datagram Sockets 4
3.1.3 Selection of Socket Type 5
3.1.4 Addresses and Ports 6
3.1.5 Socket Creation and Connection Maintenance 6
3.1.6 Data Transfer 7
3.1.7 Typical TCP Socket Session 7
3.1.8 Typical UDP Socket Session 8
3.2 IEEE-488 Control Interface 8

4. SAMPLE APPLICATIONS 10
4.1 Sample Application Overview 10
4.2 DOS Implementation Approach 11
4.3 Windows Implementation Approach 12
4.4 OS/2 Implementation Approach 13
4.5 Recommendations 14

APPENDIX A - MS-DOS Libraries 15

APPENDIX B - Windows Libraries 18

APPENDIX C - OS/2 Libraries 20

APPENDIX D - Diskette Contents 23

P148599.PDF [Page: 9 of 35]

1. INTRODUCTION

This report describes an Ethernet software development toolkit created by Software Kinetics to support development of real-time programs for the Advanced Communications ESM System (ACES). The toolkit supports three PC based operating systems/environments; MS-DOS, Microsoft Windows, and OS/2. It provides support for communications between workstations using an Ethernet LAN, and command and control of remote instruments (receivers, spectrum analysers etc.) using an IEEE-488 interface.

The remainder of this report is organized as follows. Section 2 lists reference manuals and publications. Section 3 provides an overview of the Ethernet Interface and the IEEE-488 interface. Section 4 describes the implementation approach and sample application development under each of the environments. A detailed description of toolkit components for the MS-DOS, Windows and OS/2 environments are described in Appendices A, B and C respectively. Appendix D describes the contents of the diskettes containing the Ethernet Toolkit Software.

1.1 Background

ACES consists of a number of workstations connected via an Ethernet LAN. The majority of these workstations are SUN based, but some PC-based workstations are required to run some applications. Certain ACES components are required to interface to the Data Fusion And Correlation Techniques Testbed (DFACTT).

The DFACTT system uses the TCP/IP protocol suite to transfer information between workstations, and uses the Berkley Socket abstraction as an interface to the application software. DFACTT workstations and servers communicate using both connection oriented TCP sockets and datagram oriented UDP sockets. In order to permit communications between the ACES sensor processing workstations and DFACTT, the ACES workstations will require TCP/IP support libraries. These are included in the SUN based UNIX platforms, but the PC based workstations do not include support for TCP/IP as part of the operating system.

Three operating systems are commonly used with Intel 80x86 based PCs; MS-DOS, Microsoft Windows, and IBM OS/2. TCP/IP support is available from a number of vendors for each of these operating systems. A number of these packages were evaluated, and the following were chosen for use with the Ethernet Toolkit; FTP Software's PC/TCP for MS-DOS and Microsoft Windows, and IBM's TCP/IP for OS/2 was selected for OS/2. The network communications support provided by the Ethernet Toolkit is based on these packages.

P148599.PDF [Page: 10 of 35]

ACES PC based workstations also require the ability to control IEEE-488 based devices such as spectrum analysers, receivers, signal generators etc. IEEE-488 support is not included with any of the PC operating systems, and a number of IEEE-488 hardware interface cards and support drivers are available. The National Instruments AT-GPIB interface was chosen for use during development of the Ethernet Toolkit. A single driver package supports both DOS applications and Microsoft Windows, a second driver package was acquired for OS/2.

P148599.PDF [Page: 11 of 35]

2. REFERENCES

The following software manuals form an integral part of the Ethernet Toolkit:

[1] PC/TCP Development Kit for DOS, Version 2.2, FTP Software, North Andover, Mass. U.S.A.

[2] PC/TCP for DOS, Version 2.2, FTP Software, North Andover, Mass. U.S.A.

[3] TCP/IP Version 1.2.1 for OS/2 (Refresh): Users Guide, Fourth Edition, IBM Corporation, Research Triangle Park, North Carolina, U.S.A.

[4] Getting Started with Your AT-GPIB and the NI-488.2 MS-DOS Handler, August 1990 Edition, National Instruments Corporation, Austin, Texas, U.S.A

[5] NI-488.2 MS-DOS Software Reference Manual, February 1991 Edition, National Instruments Corporation, Austin, Texas, U.S.A.

[6] Using your GPIB Software with Microsoft Windows, January 1991 Edition, National Instruments Corporation, Austin, Texas, U.S.A.

[7] Getting Started with Your AT-GPIB and the NI-488 Software for OS/2, September 1992 Edition, National Instruments Corporation, Austin Texas, U.S.A.

[8] NI-488 OS/2 Software Reference Manual, March 1990 Edition, National Instruments Corporation, Austin, Texas, U.S.A.

[9] Internetworking with TCP/IP Volume 1, Second Edition. Prentice Hall, Englewood Cliffs, New Jersey, U.S.A.

P148599.PDF [Page: 12 of 35]

3. INTERFACE DESCRIPTION

3.1 Ethernet Communications Interface

The Ethernet Toolkit supports inter-computer communication using the Berkley (UNIX) Socket abstraction. A socket is an endpoint for communication that can be named and addressed in a application that will operate using a network. From the application's perspective, it is a resource allocated by the operating system that is represented by an integer called a socket descriptor. Once this descriptor has been created and a connection is established with a peer application, data is transferred using read and write calls to the socket interface, much in the same way that an application program would read and write to a file.

The socket interface was developed to provide applications with a network interface that hides the details of the physical network. The interface is differentiated by the the two services that are provided; both Stream and Datagram sockets are available, each providing a specific type of service to an application. The two types of sockets are briefly described below. A detailed description of the socket interface can be found in reference [9].

3.1.1 Stream Sockets

The Stream Socket interface defines a reliable connection oriented service between computers. Data is transferred error free without duplication and is received in the same order that it is sent. Flow control is built in to avoid data overruns. No boundaries are imposed by the network on the data; it is considered to be an uninterrupted stream of bytes. An example of an application that uses Stream Sockets is the file transfer program FTP.

A stream socket, once established, provides a transparent path for data transfer between workstations. The application is guaranteed that the data sent into the stream will appear at the destination in the correct order. Stream sockets are also known as TCP sockets.

3.1.2 Datagram Sockets

The Datagram Socket interface defines a connectionless service. Datagrams are sent as independent packets. The service provides no guarantees; data can be lost or duplicated, and datagrams can arrive out of order. The size of a datagram is limited to the size that can be sent in a single transaction (this value is implementation specific). No disassembly and reassembly of packets is performed. An example of an application that uses datagram sockets is the Network File System (NFS).

The Datagram Socket interface can be extended by defining new socket types to provide

P148599.PDF [Page: 13 of 35]

additional services. Error detection and correction protocols (simple or complex) can be added on top of a Datagram Socket to improve reliability and accuracy. This is not normally required in a small local area network, unless large amounts of data are being transferred and buffer allocation is problematic. Datagram sockets are also known as UDP sockets.

3.1.3 Selection of Socket Type

The two socket types provide very different types of services. It is important to consider a number of factors when deciding which type of socket to use for an application. These factors are listed below:

1) Reliability

Stream sockets provide the most reliable data transfer path. Datagram sockets are unreliable because packets can be discarded, corrupted or duplicated during transmission. This may be acceptable if the application does not require reliability, is in a well defined environment (i.e. a small LAN) or if the application implements the reliability on top of the sockets interface. The tradeoff is the increased performance over Stream Sockets.

2) Performance

The overhead associated with reliability, flow control, packet reassembly and connection maintenance degrade the performance of stream sockets to the extent that they do not perform as well as datagram sockets in many applications.

3) Transfer Requirements

Datagram Sockets impose a limit on the amount of data that can be transferred atomically. If the application sends less than approximately 2048 bytes at a time, Datagram Sockets are more efficient. As the amount of data sent during a single transaction increases, it makes more sense to use Stream Sockets.

These three factors must be carefully evaluated before designing any application that will use sockets. In addition, the concept of a fixed connection between client and server (which is required when using a stream socket) must be incorporated into the application program architecture if stream (TCP) sockets are used.

P148599.PDF [Page: 14 of 35]

3.1.4 Addresses and Ports

Each host interface has a unique Internet Address. These are 32 bit quantities that select a specific host/interface. The 32 bit address is split into four 8 bit segments using a "." between each value (e.g. 131.136.32.150). Data transfer paths between hosts are further differentiated using ports. A ports is represented by a 16 bit value that provide an additional qualifier used by the system software to get data to a specific application (an address could be listening to a specific port for connection requests or data). Some ports are reserved for particular applications and are known as well known ports (e.g. Telnet connections use port 23). Port designators above 1024 are generally available and utilized by user applications.

When a socket is created it can be associated with an specific port and address. This is known as "binding" a socket to an address. This is required when a socket is to be used as a server, so that it will be accessable to clients at a particular address and port number. UDP sockets can be bound to an address of zero (0.0.0.0), which will allow data to be received from any address.

3.1.5 Socket Creation and Connection Maintenance

Sockets are created through an operating system call to the socket interface. This is usually in the form of a "socket" call, which will allocate a socket descriptor from the pool. The descriptor is generally a return argument and is used for all other operations on the socket. Once a socket is created, it is generally bound to an address and port as described above.

Connection oriented (TCP) sockets require a certain amount of overhead related to establishing and breaking the logical connection. The following operations are used:

Listen - the listen call is used by a server to prepare a socket for incoming connections. In terms of underlying protocols, listen puts the socket in a passive mode ready to accept connections.

Connect - the connect call is used by a client to attempt to establish a logical connection with a server. If a server is available at the selected port and address, the call will return with a connection established and data transfers may take place. If no server is available, the call will time out and return with an error code.

Accept - the accept call is a blocking call that is used by the server application to wait for a connection. When a connect request is received, the operation is completed and a new socket is returned that is connected to the client. This new socket descriptor is then used for data transfers with the client.

Close - the close call is used by the client or server to break the connection.

P148599.PDF [Page: 15 of 35]

3.1.6 Data Transfer

Once a connection has been established (TCP sockets) or a socket has been created and bound to an address (UDP sockets), data transfers may take place. Connection oriented (TCP) sockets use "send" and "recv" calls to write to and read from the data stream. Connectionless (UDP) sockets can use these same calls, but also may use "sendto" and "recvfrom" calls to send and receive from specific addresses. The data transfer calls available are fully described in reference [9].

3.1.7 Typical TCP Socket Session

A typical TCP socket session is depicted in Figure 3.1.7-1, showing the relationships between the client and server operations, and the sequencing of events.

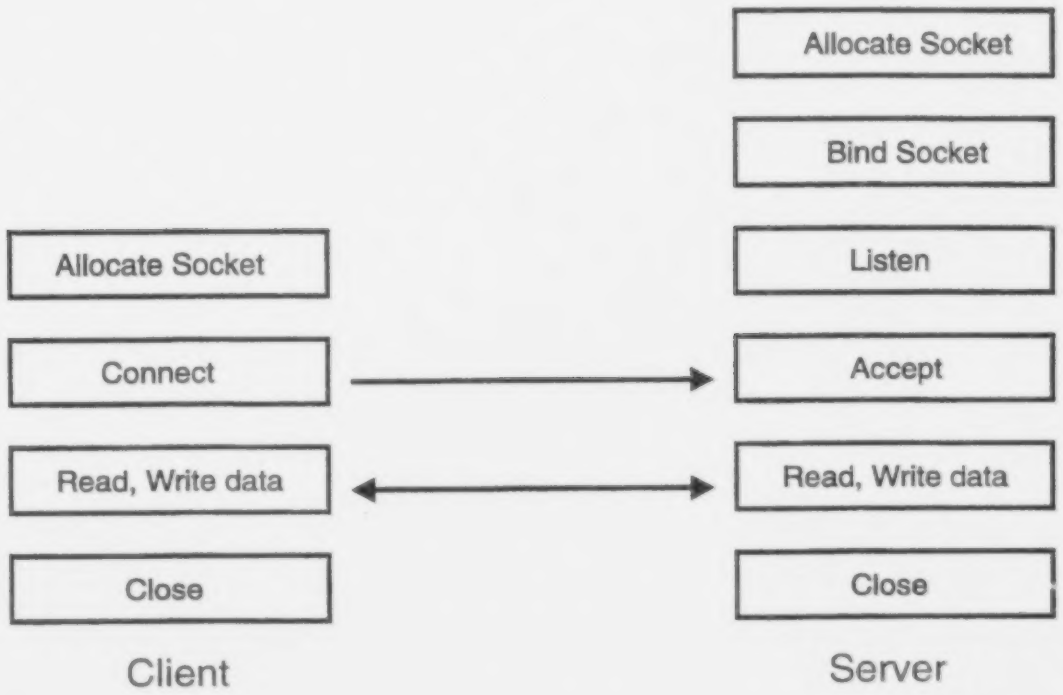


Figure 3.1.7-1
TCP Socket Session

P148599.PDF [Page: 16 of 35]

3.1.8 Typical UDP Socket Session

A typical UDP socket session is depicted in Figure 3.1.8-1, showing the relationships between the client and server operations, and the sequencing of events.

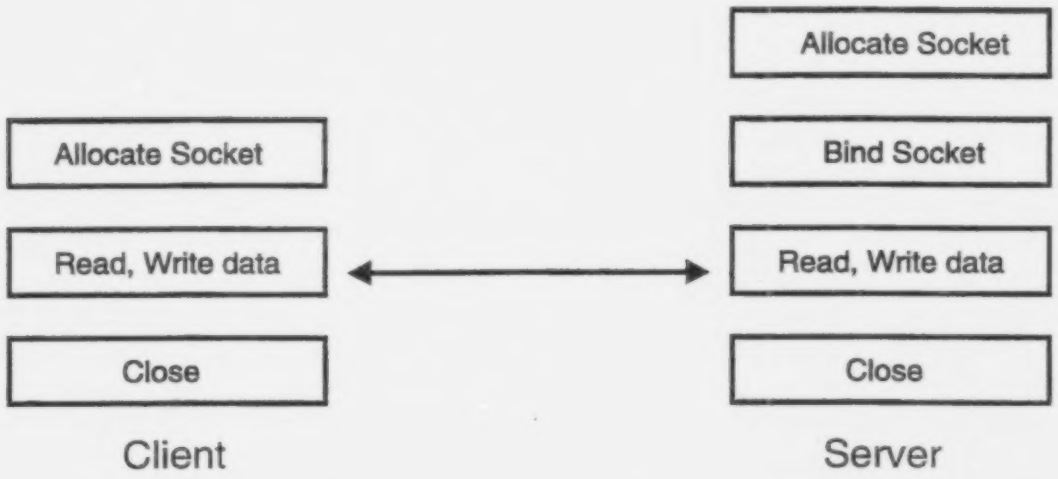


Figure 3.1.8-1
UDP Socket Session

3.2 IEEE-488 Control Interface

The IEEE-488 standard defines a link, or interface system, that is used to provide command, control and data transfer between devices on a bus. The original bus was defined by Hewlett Packard (where it is called the HP-IB) to connect and control programmable instruments manufactured by HP. It supports data transfer rates of up to 1 megabyte/sec, and quickly gained popularity in other applications such as intercomputer communications and peripheral control. This popularity led to the formalization of a standard definition by the Institute of Electrical and Electronic Engineers (IEEE-488). It is now widely known as the General Purpose Interface Bus (GPIB).

The GPIB carries device dependent messages and interface messages. Device dependent messages, often called data or data messages, contain device specific information such as programming instructions, measurement results, machine status, and data files.

P148599.PDF [Page: 17 of 35]

Interface messages manage the bus itself. They are usually called commands or command messages. Interface messages perform such tasks as initializing the bus, addressing and unaddressing devices, and setting device modes for remote or local programming. The term command as used in this context should not be confused with some devicd instructions which can also be called commands. Such device-specific instructions are actually data messages.

There are three roles for devices that are connected to the GPIB; Talker, Listener and Controller. A Talker sends data messages to one or more Listeners. The Controller manages the flow of information on the GPIB by sending commands to all devices.

THe Controller addresses a Talker and Listener before the Talker can send its message to the Listener. After the message is transmitted, the Controller may unaddress both devices. Some bus configurations do not require a Controller. For example, one device may always be a Talker (called a Talk-only device), and there may be one or more Listen-only devices. A controller is necessary when the active of addressed Talker or Listener must be changed. The Controller function is usually handled by a computer.

The GPIB board used by the Ethernet Toolkit is manufactured by National Instruments. It is a 16 bit peripheral card known as the AT-GPIB. It has the ability to play any of the three roles; Controller to manage the GPIB, Talker to send data to a peripheral, and Listener to receive data. The software drivers provided with the card for DOS, Windows and OS/2 automatically determine which role is required (e.g. Talker when data is sent through a write call), and switching between roles is transparent to the application program.

The GPIB uses 16 signal lines. Eight of these are used for data transfer to support bytewide communications. Three lines are used for handshaking between devices (data transfers are asynchronous). The remaining five lines are used for interface management to manage the flow of information across the interface.

P148599.PDF [Page: 18 of 35]

4. SAMPLE APPLICATIONS

4.1 Sample Application Overview

In order to test the libraries developed for each of the three operating systems, a sample receiver control application was created to exercise the Ethernet and GPIB interfaces. Two PC's were used to transfer data back and forth using an ethernet link, and each PC had a National Instruments GPIB interface installed. The application code running on each PC was designed to read the parameters of a local receiver and display them on the screen. A second display pane was created on each PC to show the remote receiver parameters. This required each copy of the application to monitor and update the local receiver parameters, transmitting any changes to the remote PC through the ethernet connection. The display panes were to be interactive in that the user could change the receiver parameters either from the receiver front panel or from the PC display. The nature of the application indicated that the most effective use of the Ethernet would be as a connectionless UDP link. A third requirement was to continuously display time of day to simulate a background activity.

Interoperability between different application environments was ensured through the use of a standard message format to transfer receiver parameters between PCs. The following C structure was used (translated to Smalltalk for the V/Windows application):

```
typedef struct wj_msg {
        BYTE msg_size;
        BYTE local;
        char freq[FREQ_SIZE];
        BYTE modn;
        BYTE bw;
        BYTE checksum;
        } wj_msg;
```

Whenever a receiver's parameters were changed, the application was required to send the wj_msg to the other PC to allow its display windows to be updated. The "local" parameter was used to determine if the updated parameters applied to the local receiver or remote receiver.

The following sections briefly describe the approach taken to implement the sample application under each of the operating systems/environments. The reader is referred to the source code on the corresponding diskette (see Appendix D) for details regarding each implementation.

P148599.PDF [Page: 19 of 35]

4.2 DOS Implementation Approach

The DOS implementation of the receiver control application was developed as text mode application that uses BOSS windows to provide a character based windowed user interface. The application code was written in "C", using Borland C/C++ for DOS v3.1. Source code is included on Disk #1 in the DOSAPP directory.

The lack of multi-tasking support in DOS resulted in the use of a continuous loop to perform all required processing. This loop performed the following functions:

- check for input from the operator
- check for input from the local receiver interface
- check for input from the data socket
- update the time display on the screen

Separate modules were used to process any input received during the polling loop. A special event handler was created to receive incoming UDP data packets. The display window is shown in Figure 4.2-1.

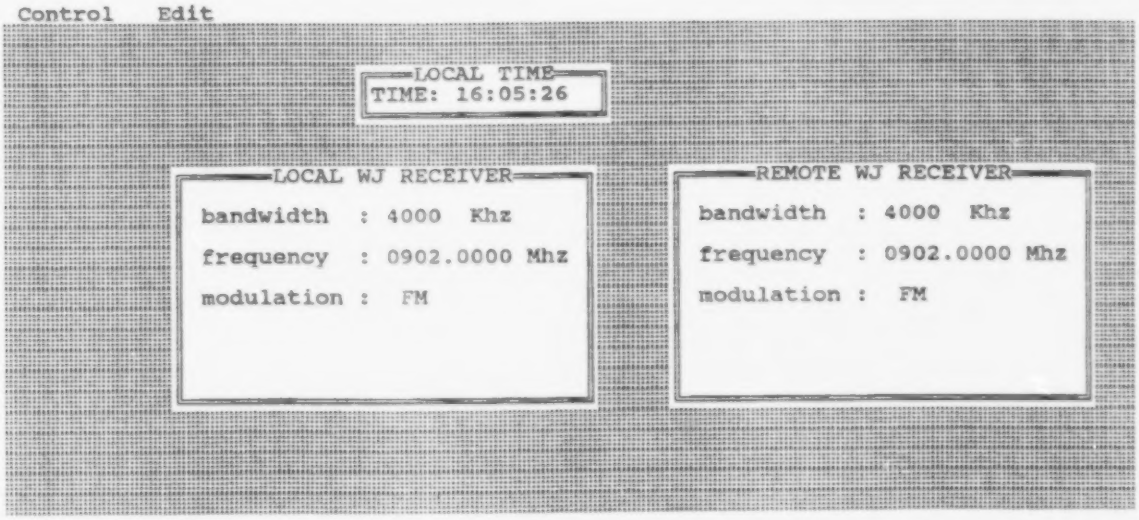


Figure 4.2-1
DOS Application Display

P148599.PDF [Page: 20 of 35]

4.3 Windows Implementation Approach

The Windows implementation of the receiver control application was developed as true Windows application using Smalltalk V/WIndows. The rapid prototyping features of Smalltalk were used to quickly develop a basic user interface, which was then enhanced to provide required functionality. The display windows includes basic Windows display tools; sliders, radio buttons and spin buttons. The source code for the Smalltalk application is included on Disk #1 in the WINAPP directory. The display window is shown in Figure 4.3-1 below.

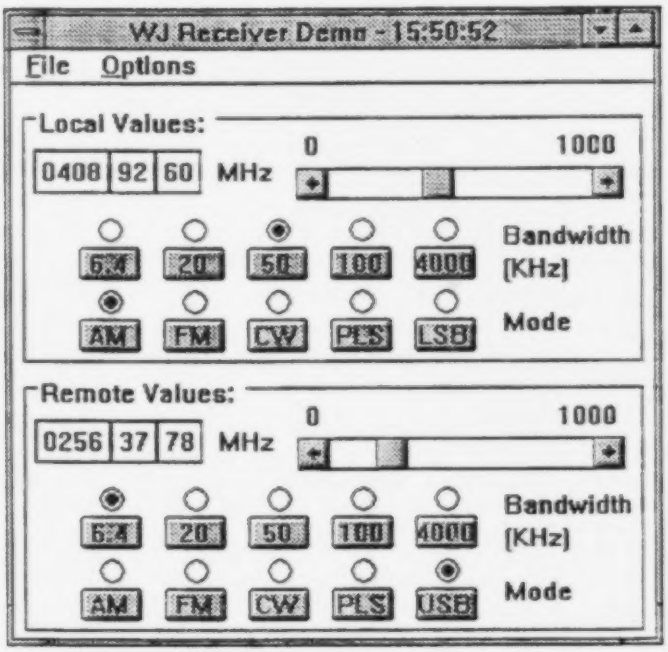


Figure 4.3-1
Windows Application Window

P148599.PDF [Page: 21 of 35]

4.4 OS/2 Implementation Approach

The OS/2 implementation of the receiver control application makes full use of the advanced control mechanisms provided by the OS/2 Presentation Manager, and the multi-thread processing model provided by the OS/2 kernel. The application code was developed using Borland C for OS/2. The IBM TCP/IP support library was customized to interface with the 32 bit C compiler (the current IBM distribution contains a 16 bit interace specific to IBM's C/SET 2).

The multi-thread support is used in the application to simplify programming and provide a high level of performance without sacraficing system resources. The use of threads is full described in the application source code, which is included on DISK #1 in the OS2APP directory. The user interface includes some of the more common Presentation Manager tools; sliders, radio buttons and spin buttons. The display window is shown in Figure 4.4-1

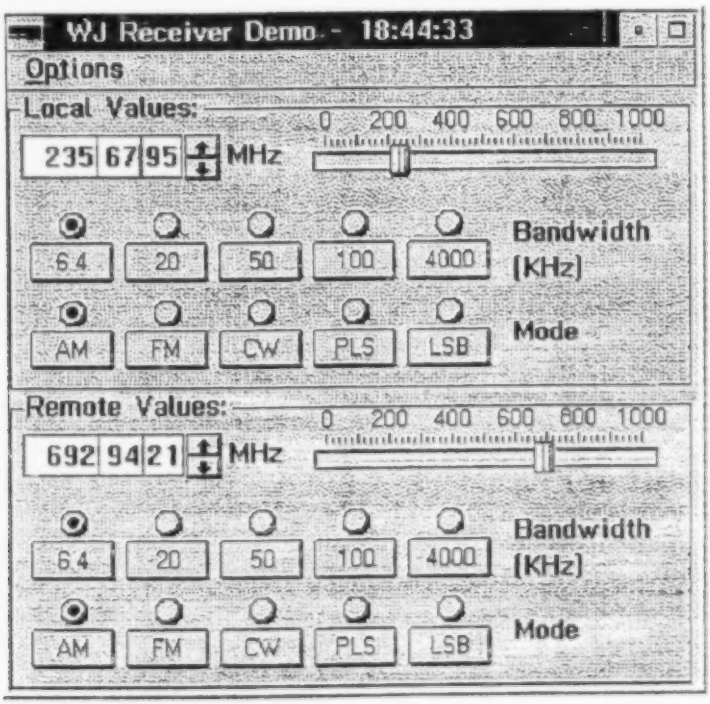


Figure 4.4-1
OS/2 Application Window

P148599.PDF [Page: 22 of 35]

4.5 Recommendations

Development of the Ethernet Toolkit and the sample WJ Receiver control application has highlighted the strengths and weaknesses of the three development environments for the PC. These differences are outlined below.

MS-DOS

The MS-DOS environment is the simplest of the three environments used during the course of the project. DOS is a simple program loader and file manager that supports basic disk, serial and parallel I/O. The BOSS windows package provides a character based window environment that is suitable for basic applications that do not require graphics display and can be controlled from the keyboard. The FTP PC/TCP package supports all normal network applications (Telnet, FTP etc) and also supports development of applications using a Berkley Socket Library. The two flaws with the DOS environment are the lack of multitasking capability and the absence of a graphical interface. It does, however, provide basic support that relies on the target language (i.e. Borland C) for graphics and BOSS Windows for a simple character based window user interface.

Windows

Microsoft Windows is a graphical user interface on top of MS-DOS. It supports task switching between multiple applications, and an easy to use mouse based user interface. It is the most popular PC environment at this time primarily due to the graphical user interface, but is limited in it real-time capabilities and has no multi-tasking support. It is also limited by the segmented architecture of the 80x86 processors. It is most suitable for graphical, non real-time applications.

OS/2

The OS/2 operating system addresses the deficiencies of both DOS and Windows, in addition to supporting programs targetted for both of these environments. The two most powerful features are the flat 32 bit memory model and direct support of multitasking by the operating system. This simplifies programming of complex real-time applications by allowing operations that are independent and concurrent to be implemented as separate threads, which are allocated processor time as required by the operating system.

In summary, this report recommends continued use of two of the three operating environments - MS-DOS and OS/2. MS-DOS can be used for simple applications that do not require multiple tasks and mouse driven user interfaces. For more complex applications, the multi-tasking capabilities present in OS/2 combined with the flat memory model make it the platform of choice for larger, complex applications.

P148599.PDF [Page: 23 of 35]

APPENDIX A - MS-DOS Libraries

A.1 TCP/IP Socket Library for DOS Borland C++

The TCP/IP Socket Library is based on a TCP/IP support package produced by FTP Software of Wakefield, Mass. This package supports many different network cards, a number of different networks (X.25, Ethernet, SLIP, etc.) and includes a suite of support tools (Telnet, FTP, LPR etc.). Support for program development is provided by the PC/TCP Development Kit for DOS, also supplied by FTP Software.

Use of the DOS socket library requires installation of the memory resident drivers and linking of a number of libraries to the application program. The components shown in the diagram are described below.

1) WD8003 Interface Card

The WD8003 Interface Card provides a PC compatible link to either thin or thick Ethernet. The card comes with a setup program (EZSETUP.EXE) that allows the user to configure the card to specific interrupt, I/O address and memory buffer address locations. Note that the WD8003 packet driver described below also supports the 16 bit version of the interface card, the WD8013.

2) Packet Driver - WD8003.EXE

The Packet Driver is actually a card specific device driver that provides the ethernet kernel with a standard interface to transmit and receive data. The WD8003.EXE file must be called from the AUTOEXEC.BAT file and is invoked when the PC starts up. Four parameters are required; the interrupt vector, the interrupt number, the I/O address and the memory address. The interrupt vector should be set to 0x62, and the other parameters should match the configuration of the card. For example, a WD8003 card configured to interrupt 5, I/O address 300 Hex and Memory Address D800 Hex would be installed by having the following line in the AUTOEXEC.BAT file:

C:\PCTCP\WD8003.EXE 0x62 5 0x300 0xD800

The driver will confirm proper installation by displaying a message on the screen during startup.

P148599.PDF [Page: 24 of 35]

3) Ethernet Driver - ETHDRV.EXE

The Ethernet Driver layer provides an ethernet compatible kernel supporting all low level network functions. The ETHDRV.EXE file must be called from the AUTOEXEC.BAT file and is invoked following the WD8003.EXE file. No parameters are required. The kernel will confirm proper installation by displaying a message on the screen during startup.

4) Configuration - PCTCP.INI

The configuration file, PCTCP.INI is read by the kernel during startup. The configuration includes a number of network parameters, including the host internet address, name and other user specific data. It is created during the installation of the PC/TCP package and is stored as an ASCII file complete with comments and parameter descriptions, and can be edited by the user if required following installation. Note that the file name should be included in the DOS environment by setting the PC/TCP environment variable to the file pathname. This is done through the AUTOEXEC.BAT file before the ethernet kernel is installed. If the configuration file were stored in the C:\PCTCP directory, the following command would be include in the AUTOEXEC file:

SET PC/TCP=C:\PCTPC

5) Network Interface - LNETLIB.LIB

The network interface library includes the basic network independent functions that manage connections and data transfers.

6) Host Library - LPC.LIB

The Host Library includes all functions that are host specific and related to network byte ordering. This includes functions to encode and decode host addresses and port numbers. This file must be linked with the application code if any of these utilities are being used.

7) TCP Interface - WJTCP.LIB

Additional socket interface code to tie the asynchronous call handler to the socket library. The asynchronous call handler allows the application to be notified (by checking a flag variable) when data is received from a socket.

P148599.PDF [Page: 25 of 35]

8) Async Call Handler - WJDATA.ASM

The low level interrupt handler is used to process the event generated by the socket library when data is received.

Applications that intend to use the socket library must be compiled with the following libraries and object files: WJDATA.OBJ, WJTCP.LIB, LPC.LIB, LSOCKET.LIB, LNET.LIB. Note that these libraries have been compiled for the LARGE programming model, and the application must be developed using this model.

A.1.1 Installation and Configuration

The installation and configuration of the PC/TCP package is described completely in reference [2]. A key point to remember is that the PCTCP.INI file must be modified using a file editor, and the system must be rebooted for the new parameters to take effect. In addition to the socket library interface, the PC/TCP package includes a full suite of utilities including Telnet, FTP, Finger etc.

A.2 DOS IEEE-488 Support

The NI-488 for DOS handler is contained in a single device driver file; GPIB.COM. Installation of the driver is accomplished by copying the GPIB.COM file from the National Instruments distribution disks to a driver directory on the target system. To activate the driver, the following line must be placed in the DOS CONFIG.SYS file:

device = c:\<driver directory name>\gpib.com

Note: the system must be rebooted in order for the driver to become active.

Once the driver is installed, it must be configured to match the jumper settings on the interface card and the names and characteristics of the GPIB devices to be used by the application. This is accomplished by through the configuration utility, which is contained in the file IBCONF.EXE on the distribution disk. IBCONF is a screen oriented, interactive program tha is supplied allows the user to view or change the configuration of the GPIB.COM driver. Configuration information can be viewed or changed in a configuration file, in the memory resident copy of the handler, or in a disk copy of the hander. Further information regarding modification of the GPIB.COM parameters can be found in Section 2 of Reference [5]. The application interface also described in this manual.

P148599.PDF [Page: 26 of 35]

APPENDIX B - Windows Libraries

B.1 Windows TCP/IP Support

The TCP/Library for Smalltalk/V WIndows is a combination of the PC/TCP support for Windows (a Dynamically Linked Library or DLL) and a number of Smalltalk classes developed to interface to the PC/TCP DLL. The class structure is very similar to the V/286 Socket interface used by OTI. The classes are described individually below. Detailed descriptions regarding use of the classes can be found in the PC/TCP Development Kit for DOS (reference [1]), and in the sample application source code on Disk #1.

Class	Description
NetworkAddress	Class used to create and translate network addresses for use with send, recv and bind calls
SocketAddress	Low level address manipulation code.
Socket	Socket descriptor class. Includes subclasses for TCPSocket and UDPSocket. Provides basic protocol to create, bind, listen, connect, accept and transfer data between sockets.
SocketDLL	Low level interface to the PC/TCP socket DLL.
WSAData	Class used to extract varables from the DLL's WSAData data structure.
WSAAddress	Class used to store socket addresses.

B.1.1 Installation and Configuration

The PC/TCP support for MS Windows is based on the DOS support structure described in Appendix B. Two additional DLLs are used to provide an interace to the DOS functions; SOCKET.DLL contains the standard socket interface routines (accept, socket, send etc). A second DLL, PCTCPAPI.DLL contains the interface to the DOS functions and is called by the SOCKET.DLL functions. Both of these DLL files must be copied from the installation directory (normally C:\PCTCP) to the working directory in order for the application code to execute correctly.

P148599.PDF [Page: 27 of 35]

B.2 Windows IEEE-488 Support

The IEEE-488 Library for Smalltalk/V WIndows is a combination of the National Instruments AT-GPIB support for Windows and a number of interfaces classes developed by Software Kinetics to interface Smalltalk to the AT-GPIB DLL. The classes interface directly to the Windows Dynamic Link Library (DLL). Each class is described below. Futher details can be found in the Smalltalk source code included on Disk #1.

Class	Description
GPIBDLL	Low level interface to the windows DLL. Includes calls to ibfind, ibread and ibwrite functions.
GPIBDevice	Class used to create a device interface (an device descriptor is an instance of GPIBDevice). Provides protocol to initialize, open and read and write to a device.
GPIBibsta, GPIBiberr, GPIBibcntl	Classes to translate status , error and control flags.

B.2.1 Installation and Configuration

The AT-GPIB support for MS-Windows is in the form of a DLL, and does not require formal installation beyond copying the GPIB.DLL file from the National Instruments distribution diskettes into an appropriate directory. Configuration of the DLL interface is accomplished through the WIBCONF.EXE program (supplied on the distribution diskettes). It is similar to the DOS based configuration program, and allows the user to change hardware parameters (address, interrupt and DMA addresses), as well as device specific parameters. The WIBCONF program must be run prior to using the Smalltalk classes to ensure the hardware and device parameters match the AT-GPIB board.

P148599.PDF [Page: 28 of 35]

APPENDIX C - OS/2 Libraries

C.1 TCP/IP Support

TCP/IP support is provided by the IBM TCP/IP version 1.2.1 for OS/2. This package provides full TCP/IP support including standard utilities such as Telnet, FTP, Mail. It also includes a number of application support libraries, including a Socket interface for OS/2 Applications. The socket interface follows the Berkley definition quite closely, and the reader is referred to reference [8] for a complete description of the interface. Both TCP and UDP sockets are supported, and when combined with the multi-threading capabilities of OS/2, form a very powerful toolset for application development.

The socket library, as supplied by IBM, was created to support IBM's CSET/2 compiler. In order to support the Borland C/C++ OS/2 compiler, it was necessary to recompile the library interface source code to support 32 bit calls. A full 32 bit interface will be released with version 2.0 of the IBM TCP/IP package in Fall of 1993, at which time the custom library can be discarded. In the interim, the following calls have been translated to 32 bit format and included in the socket library described below:

```
int     accept( int, void *, int *  );
int     bind(int, struct sockaddr_in *, int);
int     connect( int, void *, int );
int     getpeername( int, void *, int * );
int     getsockname( int, void *, int * );
int     getsockopt( int, int, int,char * , int *);
int     listen( int, int );
int     recv( int, char * , int, int );
int     recvfrom(int, char * , int, int, void * , int *  );
int     send( short, char *, short, short );
int     sendto(int, char *, int, int, struct sockaddr_in *, int);
int     setsockopt( int, int, int, char *, int );
int     shutdown( int, int );
int     sock_cancel( short );
int     sock_init( void );
int     socket(int, int, int);
int     soclose(int);
```

Applications using the socket library must include the following header files (included on Disk #2 in the OS2SUP directory) in any modules that use sockets:

```
#include "utils.h"
#include "in.h"
#include "socket.h"
```

P148599.PDF [Page: 29 of 35]

In addition, the "socket.lib" file (also on Disk #2) must be included as a library when linking an application that wishes to use the socket library.

C.1.1 TCP/IP Library Structure

The TCP/IP library is based on the 16 bit version of the library distributed by IBM, with the addition of a 32 bit interface (as described in section C.1). The modified interface source code has been included on Disk #2 should any further modifications be required. A description of the files is included below. The Borland C compiler project file (SOCKET.PRJ) has been included to simplify recompilation if required.

Library File	Description
GENIO.H	C header file - device I/O definitions
IN.H	C header file - internet constants
INET_ADD.C	C source code - internet address translation utilities
INET_NTO.C	C source code - internet address translation utilities
IOCTL.H	I/O control function definitions
NTOH.ASM	ASM source code - internet address translation utilities
SOCKET.H	Header file for socket library
SOCKET.LIB	Borland C socket library
SOCKET.PRJ	Borland C project file to build socket library
SOCKETS.C	Borland C socket library source code
UTILS.H	C header file - address translation definitions

C.2 IEEE-488 Support

IEEE-488 support is provided by the National Instruments NI-488 for OS/2 handler. This handler supports the OS/2 Application Program Interface (API) file I/O and control functions, or standard I/O functions of OS/2 API compatible languages. The OS/2 API functions are directly accessible from high-level languages; no language interface is required for C, BASIC, FORTRAN and Pascal applications.

For Borland C applications, the standard OS/2 file I/O functions are used. To communicate with a device the following I/O calls are used:

```
wjDevice = open ( "WJ-1", O_RDWR);

write (wjDevice, buffer, buffer_length);

bytes_read = read (wjDevice, buffer, max_buffer_size);

close (wjDevice);
```

P148599.PDF [Page: 30 of 35]

Input received by the IEEE-488 interface card is buffered internally until a "read" call is issued. The bytes_read value will indicate how many bytes were read, up to a maximum of "max_buffer_size".

Figure C.2-1 shows the Further information regarding the OS/2 file I/O interface can be found in the Boriand C++ for OS/2 reference manuals.

C.2.1 Installation and Configuration

The NI-488 for OS/2 handler is contained in a single device driver file; GPIB.SYS. Installation of the driver is accomplished by copying the GPIB.SYS file from the National Instruments distribution disks to a driver directory on the target system. To activate the driver, the following line must be placed in the OS/2 CONFIG.SYS file:

device = c:\<driver directory name>\gpib.sys

Note: the system must be rebooted in order for the driver to become active.

Once the driver is installed, it must be configured to match the jumper settings on the interface card and the names and characteristics of the GPIB devices to be used by the application. This is accomplished by through the configuration utility, which is contained in the file IBCONF.EXE on the distribution disk. IBCONF is a screen oriented, interactive program tha is supplied allows the user to view or change the configuration of the GPIB.SYS driver. Configuration information can be viewed or changed in a configuration file, in the memory resident copy of the handler, or in a disk copy of the hander. Further information regarding modification of the GPIB.SYS parameters can be found in Section 2 of Reference [8].

P148599.PDF [Page: 31 of 35]

APPENDIX D - Diskette Contents

The following directories and files are included on Ethernet Toolkit Disk #1:

Directory:	DOSAPP	DOS Application Files
Files:	WJ.C	DOS application source code
	WJMSG.H	Header file for DOS application
	WJ.OBJ	Object code for WJ.C
	WJ.PRJ	Borland C/C++ project file for
	WJ.EXE	Executable DOS application

Directory:	DOSSUP	DOS Support Files
Files:	WJTCP.C	Event handler interface source code
	WJDATA.ASM	Asynchronous upcaller hander source code
	MCIB.ASM	C interface to GPIB.COM TSR
	WJTCP.OBJ	Object code for WJTCP.C
	WJDATA.OBJ	Object code for WJDATA.ASM
	MCIB.OBJ	Object code for MCIB.OBJ

Directory:	WINAPP	Windows Application Files
Files:	WSADATA.CLS	WSADATA Class
	CHANGE.LOG	V/Windows change log file
	SOURCES.SML	V/Windows Smalltalk source code
	V.EXE	V/Windows executable with application code
	INITSIZE.MTH	initSize method for class DialogTopPane
	SPPRTDEV.MTH	supportedEvents method for class TopPane
	WMTMRWTH.MTH	wmTimer:with: method for class TopPane
	WMVSCRLL.MTH	wmVScroll: with: method for class TopPane
	WMKLLFCS.MTH	wmKillfocus: with: method for class SubPane
	WJADDRSS.CLS	WJAddressPrompter class source code
	WJDIALOG.CLS	WJDialog class source code
	WJRECEVR.CLS	WJReceiver class source code
	GPIBIBST.CLS	GPIBibsta class source code
	GPIBIBRR.CLS	GPIBiberr class source code
	GPIBBCNT.CLS	GPIBibcntl class source code
	GPIBDLL.CLS	GPIBDLL class source code
	GPIBDEVC.CLS	GPIBDevice class source code
	NTWRKADD.CLS	NetworkAddress class source code
	WSANAMLN.CLS	WSANameLength class source code
	WSAHSTNT.CLS	WSAHostEntry class source code
	WSADDRSS.CLS	WSAAddress class source code
	SOCKTDLL.CLS	SocketDLL class source code
	SCKTADDR.CLS	SocketAddress class source code
	SOCKET.CLS	Socket class source code

P148599.PDF [Page: 32 of 35]

The following directories and files are included on Ethernet Toolkit Disk #1:

Directory :	OS2APP	OS/2 Application Files
Files:	WJ.DEF	Definition file for application
	WJ.RES	Resource file for application
	WJ.C	C source code for application
	WJ.H	C header file for application
	WJ.ICO	OS/2 desktop icon for application
	WJ.MAK	Borland C Makefile for application
	WJ.RC	Complied resources for application
	WJ.MAP	Map file for application
	WJ.EXE	Executable application code
	WJRCVR.DLG	Dialog definition for WJ receiver pane
	WJRCVR.RES	Resource file for WJ receiver pane
	MAKEFILE	Borland C Makefile for application
	WJRCVR.H	C header file - receiver definitions
	MAKEFILE.DEP	Dependencies list from makefile
	WJMSG.H	C header file - message definition
	WJ.OBJ	Object file from WJ.C
	WJ.PRJ	Borland C project file for application
	WJ.DSK	Borland C context file
	WJ.CFG	Borland C configuration parameters for WJ application

Directory :	OS2SUP	OS/2 Support Files
Files:	GENIO.H	C header file - device I/O definitions
	IN.H	C header file - internet constants
	INET_ADD.C	C source code - internet address translation utilities
	INET_ADD.OBJ	C object file for INET_ADD.C
	INET_NTO.C	C source code - internet address translation utilities
	INET_NTO.OBJ	C object file for INET_NTO.C
	IOCTL.H	I/O control function definitions
	NTOH.ASM	ASM source code - internet address translation utilities
	NTOH.OBJ	Object file for NTOH.ASM
	SOCKET.H	Header file for socket library
	SOCKET.LIB	Borland C socket library
	SOCKET.PRJ	Borland C project file to build socket library
	SOCKETS.C	Borland C socket library source code
	UTILS.H	C header file - address translation definitions

P148599.PDF [Page: 33 of 35] SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Software Kinetics Ltd.
65 Iber Road
Stittsville, Ont.
K2S 1E7

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

Unclassified

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Aces Ethernet Toolkit Final Report

4. AUTHORS (Last name, first name, middle initial)

Perrin, John P.

5. DATE OF PUBLICATION (month and year of publication of document)

March, 1993

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

24

6b. NO. OF REFS (total cited in document)

9

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Report - Software Users Manual

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

Defence Research Establishment Ottawa
Shirleys Bay
Ontario

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

None.

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

W7714-02-9601

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

#1600-118-001

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

None

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

See (11)

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both offical languages unless the text is bilingual).

This report describes an Ethernet software development toolkit created by Software Kinetics to support development of real-time programs for the Advanced Communications ESM System (ACES). The toolkit supports three PC based operating systems/environments; MS-DOS, Microsoft Windows, and OS/2. It provides support for communications between workstations using an Ethernet LAN, and command and control of remote instruments (receivers, spectrum analysers etc.) using an IEEE-488 interface.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

TCP/IP

Network

Ethernet

IEEE-488

Sockets

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

Defence Research and Development Canada | Recherche et développement pour la défense Canada | Canada

Français | Contact Us | Help | Search | Canada Site
Home | Defence R&D Canada | Defence Site

Browse Indexes

Titles

Authors

Descriptors

Corporate Authors

Corporate Acronyms

Report Numbers

Search Database

Summary

It is recommended you search "Authors" and "Corporate Authors" on their respective "Browse" pages.

Click here if you are having problems displaying text (ie. text is too big, or too small) or page appears to be jumbled.

Prior Top **Document:** 6 of 7 Bottom Next Click here to Order Click here to Download

TITLE:(U) ACES ETHERNET TOOLKIT

CORPORATE AUTHOR: Software Kinetics Ltd, Stittsville ONT (CAN);Defence Research Establishment Ottawa, Ottawa ONT (CAN)

ABSTRACT: (U) Describes an Ethernet software development toolkit created by Software Kinetics to support development of real-time programs for the Advanced Communications ESM System (ACES). The toolkit supports three PC based operating systems/environments; MS-DOS, Microsoft Windows, and OS/2. It provides support for communications between workstations using an Ethernet LAN, and command and control of remote instruments (receivers, spectrum analysers etc.) using an IEEE-488 interface.

DESCRIPTORS: (U) SOFTWARE DEVELOPMENT;REAL TIME OPERATIONS;ACES(ADVANCED COMMUNICATIONS ESM SYSTEM);COMPUTER NETWORKS

IDENTIFIERS: (U) IEEE-488 INTERFACE;ETHERNET;TCP/IP

REPORT NUMBER: SKD-1600-118-001;DREO-CR-93-669; Technical Report (Final Report)

REPORT DATE: 31 Mar 1993

NUMBER OF PAGES : 32

DRDKIM N°: 95-01219

CANDIS N°: 148599

Format(s): Hardcopy;Document Image stored on Optical Disk

Date modified: 2003-04-17

Important Notices